# *Towards a real-time simulation environment on the edge of current trends*

Eugene Chemeritskiy
The Faculty of Computational Mathematics and Cybernetics, Moscow State University,
Moscow, Russia
tyz@lvk.cs.msu.su

Konstantin Savenkov (advisor)
The Faculty of Computational Mathematics and Cybernetics, Moscow State University,
Moscow, Russia
savenkov@cs.msu.su

**This paper is devoted to renewing of the simulation project that has been undoubtedly a successful one and has endured a wide range of tasks, but is slowly and inevitable getting obsolete. In attempt to stay in the top, the development of the new runtime is started taking into account some historical regularities and currents trends in the distributed real-time simulation and some adjoining areas. The paper describes the problem scope resulted from application of the considered technologies, analyzes its possible solutions and estimates the related labor cost.**

***General Terms: Simulation Runtime, Distributed Real-time and Embedded Systems, High Level Architecture.***

## I. Introduction

In the 1990s the Computer Systems Laboratory (CS Lab) at Computational Mathematics and Cybernetics department of the Moscow State University developed a parallel modeling and simulation system called DYANA [1]. This simulation system has been used a lot as a basis for researches and development of a number of specialized simulation tools. One of these tools called STAND [2] is a hardware/software environment for hardware-in-the-loop simulation of the distributed real-time and embedded systems (DRE).

The STAND environment has been applied to a number of DRE simulation projects and proved its efficiency. To remain at the same high advantageous level in the context of fast progress in the whole IT area, it was decided to construct new runtime following the current trends to standardization in the simulation fields.

The standard-compliant runtime subsystem automatically guarantees model compatibility. Models written in accordance with the standard specifications could be always executed with use of this runtime. Similarly, natively developed models could be executed by any other certified system. This compatibility could result in product popularization and the formation of user community, and a large number of users, in its turn, could accelerate the project development and lead to its further improvement.

Replacement of the STAND native runtime raises a number of problems that could be separated into the following groups in accordance with their nature.

### *A. Designing of DRE-supporting runtime in pursuance of the latest simulation trends*

Being quite a specific simulation case, DRE simulation imposes some additional requirements to the runtime. Currently, there is no any off-the-rack and well-fitted simulation standard. Thereby some adjoining simulation areas have been explored. In attempt to mark the current trends in these areas, the third section of this paper gives a brief concept of the simulation historical path and its progress regularities. For each of the adverted innovations, the application goals and prospects are described in context of the considered project development.

Once the runtime is conceptually designed, the time comes to its implementation. Despite the considered technologies are relatively new, all of them have certain users and it is possible to learn from their experience. The refinement of the existing solutions and their adaptation to the purposes of the considered project is far less labor-intensive than the development from scratch. Thereby, the paper describes some possibilities for the adoption of the turnkey solutions.

### *B. Integration to the STAND environment and maintenance of the legacy projects*

The next aspect of STAND runtime replacement is reuse of the other STAND components. STAND software package includes a number of additional assistance subsystems such as trace collector, dynamic visualizer, version control system, integrated model development environment and so on. All listed subsystems are interconnected and have certain dependencies from each other. Due to the runtime is not a rule exception, replacement of this subsystem generates a large amount of integration problems.

The integration problems are compounded by the necessity of legacy project maintenance. The STAND environment provides a highly specialized C-based model development language. This language includes some functionality to

This work was supported in part by the Ministry of education and science of the Russian Federation under Grant “Development of an integrated environment and complex analysis methods for distributed real-time computer systems functioning”.

simplify DRE simulation (e.g. integrated support of the DRE data transmitting channels). These features are often implemented as low-level functions integrated deeply inside the runtime. Because of the interface limitations imposed to the new simulation runtime by specifications of the selected simulation standard, the effective implementation of the mentioned functionality becomes a serious research challenge.

## II. The STAND simulation environment

Modern DRE systems consist of multiple devices connected by data transfer network which contain dozens of channels. Development of DRE devices and of the DRE itself is a distributed process performed by several workgroups and the device prototypes become ready for integration in different points of time. To meet the deadlines for DRE development, the integration testing operations should begin in advance, when some of the components are not implemented yet [2].

STAND enables incremental DRE gradual integration the DRE according to the schedule of incoming devices. On early stages of the DRE integration, most (or all) of the devices are represented by the simplest simulation models reflecting only a basic schedule of data exchanges. Then the detail level is gradually increased upto full-scale models that include software of real devices and generate appropriate data matching the one generated by the device prototypes. On the next step of integration the models are step-by-step replaced by real devices that perform data exchange through the real channels.

On every listed stage of the DRE integration, the available set of devices and models could be analyzed and validated. This approach provides the abilities to detect and fix existing device errors in the earliest development phases and to reduce the DRE development cost subsequently.

The considered simulation environment contains tools intended to solve the following simulation-related tasks [2]:

1. Development of simulation models of DRE devices and auxiliary synthetic simulation models (e.g. model of the external environment);

2. Support for real-time execution of the available DRE component set including the model-device interactions through hardware channels;

3. Dynamic visualization of the simulation state and results in graphical and tabulated form and abilities for human-assisted control of the simulation;

4. Recording and processing of the simulation results, interaction with hardware monitors for data exchange channels.

## III. Tracing current trends

### *A. Interface standardization*

Simulation as a method for exploration of diverse object properties and regularities among them outruns the advent of computers for many years. However, its rapid development started after the complex mathematical calculations had been assigned to fast and reliable computers. In the beginning of the 1950s, the term simulation acquired the default meaning of digital computer simulation. Subsequently the simulation was defined as a combination of designing of the observed system model and holding the necessary experiment set on digital computers [3].

The observed system here means a separated part of the world corresponding to the domain of researcher interests. This world view is isolated during the experiment and consists of a component set. Each of these components is characterized by its property set and the dynamics of their change. Such a system could exist in reality or be imagined, can receive information and/or transmit it to its environment [4].

Abstraction that holds a subset of the observed system properties is called a model. The selected property subset should meet the objectives of the simulation. The result of simulation has any sense only in case of the simulation goals were properly identified and the constructed model is adequate to these goals [4].

From the very beginning of the simulation history the observed systems always tended to be represented in deeper detail level. This tension results in the increasing size and complexity of developed simulation model. This growth required a respective performance increase from computer systems, and this fact resulted in emergence of parallel simulation systems. These systems share the simulation task across multiple computing nodes. Typically such systems were implemented locally within the organization that wanted to use it (in accordance with this classification STAND is a parallel system created in the CS Lab) [5].

The complexity of the models was not the only factor leading to computer simulation tool evolution. The scope of simulation has been growing either. After new simulation problem types appeared, the related requirements were imposed to modeling and simulation tools. For instance, distributed simulation is often required in case of joint product development when different product component are produced by a number of workgroups located in different organizations. This type of simulation intends encompassing of several geographically separated simulation systems, which in turn may consist of a single compute node, or be a parallel system. Historically, the appearance of this task type led to the creation of distributed simulation systems that provide an essential set of services to the simulation participants and ensure its consistent behavior [5].

The next and the latest commonly recognized step in the modeling and simulation tool evolution is a standardizing of the distributed system interfaces. Using of this principle results in possibility to combine among a variety of independent simulation systems and create a general model that can be handled by every distributed system corresponding to the standard specifications [6].

### *B. DRE simulation specific*

The above classification groups existing simulation tasks and tools according to node configuration of the underlying computer system. There are lots of other features that could serve as a classification criterion. The one that is important in context of this paper is a range of supported participant types: syntactic (could be completely represented by its model) or live participants (represented by external entities). Generally live simulation type is further separated into human-in-the-loop and hardware-in-the-loop simulation depending weather the experiments requires the human presence or the external entity is a fully automated one.

Hardware-in-the-loop simulation often includes a number of physical devices, which require their data to be delivered with the respect to a given period of time (deadline), as the participants. A meeting of the deadlines in such systems is a focus of the of real-time system problematic, which are defined as those systems in which a correctness of the system depends not only on the logical results of computation, but also on the time at which these results are produced. Thereby model time must be synchronized with the astronomical one when the model interacts with hardware.

A real-time application is usually comprised of a set of cooperating tasks and they need a reliable prediction of the worst-case scenario. Apart from satisfying the timing constraints, another important characteristic of real-time systems is the notion of predictability.

Real-time systems are usually classified into two categories based on the nature of deadline, namely, hard real-time systems, in which the consequences of deadline breaking may be catastrophic and soft real-time systems, in which the utility of results produced by a task with a soft deadline decreases over time after the deadline expires. Examples of hard real-time systems are avionic control and nuclear plant control. Telephone switching system and video streaming applications are examples for soft real-time systems [6].

Besides the support of hard and soft real-time simulation, the simulation system intended to be used in DRE development should interact with additional tools providing the following capabilities:

1. Verification of the DRE devices compliance to the technical specification;

2. Integrated testing and debugging of distributed DRE software;

3. Performance and robustness evaluation of the DRE architecture;

4. Scheduling of data transfers and validation of the constructed schedules.

## IV. Designing the Runtime

The High Level Architecture (HLA) is the conventional standard in the field of distributed simulation and de facto is supported by the most of non-distributed simulation tools and by the community of distributed model developers. This standard is acceptable for DRE simulation, so it was chosen as a base standard.

Despite its initial focus on distributed simulation, using the HLA standard results in some benefits in case of the parallel simulation system (the nodes are located closely) development either. The system based on this standard can become a member of the distributed simulation and supports a range of polytypic simulation models (e.g. as-fast-as-possible synthetic models and any other types supported by the HLA standard) out of the box. In addition, the operational power of utilities devoted to distributed simulation enables easy setup of parallel simulation system node set.

The HLA standard does not currently address real-time simulation and HLA compliant simulation could not require any Quality of Service (QoS) from the underlying middleware (RTI). Indeed, there are several problems that should be solved to enable it [8]:

1. No interfaces provided to specify end to end prediction requirements for federate;

2. Management of underlying operating system(s) is unavailable;

3. In distributed case, HLA supports two transportation types only: the reliable one and the best-effort one (usually encoded with the TCP and UDP network protocols) which are not suitable for real-time constraints.

These different limitations have crucial impact for real-time simulation systems where the amount and predictability of RTI overhead is an important design factor. Thereby the considered project requires development of an additional data transmitting layer with a real-time support. Fortunately, there exist a number of related standards and associated implementations. One of the most widespread standards in this domain is the OMG Data Distribution Service (DDS) [9].

The DDS standard defines a large number of QoS policies for inter-process connection. Considering the need to meet the constraints of real time, the represented project implementation should follow the HLA standard specifications in context of inter-process communication semantics and be based on DDS standard in context of data transmission protocols.

To summarize the above, the new simulation runtime is conceptually formed around the HLA simulation standard. Because of the DRE simulation requires from the runtime some extra features (such as QoS enabled connections) not specified by HLA, the additional data transmitting middleware level (specified by the DDS standard) should underlay the usual HLA middleware (RTI) and possibly extend its functionality. STAND consists of a number of computational nodes and this imposes the resulting combined middleware to be deployed on each of them.

### *A. The High Level Architecture standard*

The roots for the HLA stem from distributed virtual environments into which users, possibly at geographically distant locations, can be encompassed. The HLA standard is a conceptual heir of Distributed Interactive Simulation (DIS) [10], which is a highly specialized simulation standard in the domain of training environments, and is used mostly for military purposes. The primary mission of DIS is to enable interoperability among separated modeling and simulation systems and to allow the joint simulation with the merged systems participation.

HLA standard remains the DIS principle relevant and extends it to the idea of polytypic model merging. Thus the HLA development began in 1993 when the Defense Advanced Research Projects Agency (DARPA) designated an award for developing of an architecture that could combine all existing modeling and simulation system types into one federation providing the reuse of existing models and simulation utilities.

There are several federation types (so called proto-federations) in accordance to the encompassed participant set [11]:

1. The Platform federation type includes DIS-style training simulations (that is real-time human-in-the-loop training simulations);

2. The Joint Training federation type stands for as-fast-as-possible time-driven and event-driven simulation (e.g. command-level military trainings);

3. The Analysis federation includes as-fast-as-possible event-driven simulations such as those that might be used in acquisition decisions;

4. The Engineering federation including hardware-in-the-loop simulations with hard real-time constraints.

The standard already has a pretty reach history and several HLA versions have been published since its appearance. Most of commercial tools currently support HLA version 1516-2000 specification. Some long term projects have being developed less intensively since of their appearance before this version have been published and are still specialized in DMSO 1.3 version. The most advanced tools are compatible with the latest IEEE 1516-2010 (Evolved).

Middleware in computing terms is used to describe a software agent acting as an intermediary between different distributed processes. It is connectivity software which allows, usually, several applications to run on one or several computational nodes and to interact across a network [6].

The middleware involved in HLA is named the Run Time Infrastructure (RTI). The RTI is the software implementation of the HLA Interface Specification. It is a middleware for the proper functioning of distributed simulation in accordance with the principles and specifications from HLA standard [11].

### B. *The Data Distribution Service standard*

OMG DDS specifications set the standard of inter-process communication, which is applicable to a broad class of distributed real-time and embedded systems (DRE). The basis of DDS is a data-centric model with the publisher-subscriber architecture (DCPS). The DCPS model forms layer, which allows the integrated processes to set a typed shared data or get the latest its version. As parts of DCPS, the global data space and namespace are created. The publisher process (the one who wants to create a shared object) should make the appropriate entries in the global data and name spaces. Similarly, the subscriber process can find the proper objects in the global namespace and access to relevant data. It is important that the announcement of the need to use the shared data and its direct use are time separated, and this approach enables the quality of service connection [7].

TABLE I
RTI IMPLEMENTATIONS

| RTI | Developer | License type |
|---|---|---|
| ARTIS GAIA | University of Bologna | Open Source[1] |
| CERTI | ONERA | GPL v2 or later |
| EODiSP | P&P Software | GPL[2] |
| MAK | MAK Technologies | Commercial |
| NCWare | Nextel | Commercial |
| Portico | Portico | CDDL[3] |
| pRTI | Pitch Technologies | Commercial |
| RTI NG | Raytheon | Commercial |

[1]Full license text is available http://pads.cs.unibo.it/
[2]General Public License
[3]Common Development and Distribution License

### C. *Evaluating of a suitable turnkey RTI implementation*

There are a lot of off-the-rack RTI implementations (Table I) and this fact gives a hope to get some developments from other projects, learning from their mistakes. Thereby, it was decided to explore the area in more details. The study was conducted among the tools, satisfying (at least partially) to the following criteria:

1. The description of the architecture and principles of implementation are available;

2. The source code of the product is available.

3. The product continues to maintain and develop;

4. The implementation is used for real-time simulation;

5. The implementation is based on the DDS standard;

Most of the examined tools are commercial, and their source code is unavailable. Thereby, benefits from the use of these implementations, taken by the developers of the target simulation system, are limited to the theoretical base. For example it is known that NCWare implementation conforms to DDS standard, and this scheme corresponds to the architectural ideas founded into the basis of considered project. The study found a number of open source systems also, and it was decided to build the target simulation system on the basis of the most suitable of them.

Unfortunately, all of the listed systems have a certain drawbacks in accordance with the purposes of the submitted project. The ARTIS GAIA implementation attracts by its advanced load balancing mechanism supplementation, but the license for this product does not allow the free use of its source code (although it is stated that the project will be fully open in future) [12]. The open source project EODiSP stopped the development in 2006 [13]. Accordingly, there is no one to assist in solving of possible development difficulties encountered. Portico project RTI is implemented using Java and, due to the language specific, it is badly compatible with the real-time simulation that is a primary goal of considered project.

Thereby the best base RTI realization for the development of the considered simulation system a priori is the CERTI one. CERTI is distributed under the GPL license, continues to evolve, and is implemented in C++ (a number of extra bindings including Java, Python, Fortran and even MATLAB is currently available). In addition, CERTI could be deployed on several combinations of platforms (Windows and Linux, Solaris, FreeBSD…) and compilers (gcc, MSVS, Sun Studio, MinGW…).

### *D. CERTI*

For years, the French Aerospace Laboratory (ONERA) develops its own HLA compliant RTI called CERTI. The project started in 1996 and its primary research objective was the distributed simulation itself whereas the appeared HLA standard was the project experiment field. CERTI started with the implementation of the small subset of RTI services, and was used to solve the concrete applications of distributed simulation theory [6].

Since the CERTI project was open sourced in 2002, a large distributed simulation developer community has been formed around the project. In many ways due to contributions of enthusiasts, the CERTI project has grown from basic RTI into a toolset including a number of additional software components that may be useful to potential HLA users.

The CERTI project has always served a base for researches in the domain of distributed simulation, and a number of innovative ideas have been implemented with its use. Thus, the problem of confidential data leak was solved in context of CERTI RTI architecture, and the considered RTI guarantees secure interoperation of simulations belonging to various mutually suspicious organizations [14]. The certain interest for the considered project is a couple of application devoted to high performance and hard real-time simulation.

In spite of HLA is initially designed to support fully distributed simulation applications, it provides a framework for composing not necessarily distributed simulations. Thereby there was created an optimized version of CERTI devoted to simulation deployed on the same shared memory platform and composed simulation running on high-performance clusters [15].

Some experience could also be adopted from ONERA project on simulation of satellite spatial system. Each federate in this federation is a time-stepped driven one. It imposes an additional requirement of hard real-time: the simulation system should meet the deadlines of each step and synchronize the different steps of the different federates [16].

Despite the distribution of commercial products, the project development is still continuing in accordance with the HLA simulation standard progress. Thus, CERTI supports HLA IEEE 1516-2000 version since 2010 in addition to previous DMSO 1.3 version.

## V. WORK SCOPE ANALYSIS

During the searches of the turnkey projects, the well suitable open source RTI implementation (CERTI) was found. To meet the real-time system requirement the internal of this middleware should gain the property of predictability and an acceptable performance.

The first problem could be solved by RTI refining in according with DDS specifications. During the constructing of the considered RTI to the DDS middleware, it is important to remain the ability of usual distributed simulation. The possible solutions are to implement the optional real-time support or provide the usual RTI-internal interface to the external simulation participants whereas staying the real-time simulator inside.

Test results show that the selected RTI loses to its commercial analogues [17], and this is largely resulted from its centralized architecture. Also the centralized architecture could be a barrier during the refinement related to DDS-compliance. Devoid of the central component the federated architecture seems to be more suitable one. However, the best suitable architecture should be identified in a separate study. Nevertheless, the architectural changes are necessary and justified. Fortunately, the CERTI RTI has been already served as a basis for creating a hard real-time simulator and some experience could be learned from that project.

There is an extra problem caused by high specialization of the STAND runtime. In some cases the functionality of the current STAND runtime could not be simulated even by the combined HLA&DDS middleware, thereby it should be injected into the middleware. Integrated support of physical data transmitting channels is a good example of this case.

Besides the mere building of a new runtime, its replacement results into a number of integration problems related to other components of the STAND environment. Each subsystem provides a certain interface to the others whereas the HLA-compliant runtime has completely different interface set. The best and the easiest way to solve the incompatibility problem is a development of appropriate interface wrappers.

The history of highly specialized standards (and HLA in particular) demonstrates a certain interface steadiness. Even if the interface has changed, the modifications are usually related to the service names and signatures whereas their semantic is the same. This solution provides an ability to replace the runtime again to the better HLA-compliant one or disintegrate some other STAND subsystems into runtime independent separated projects.

Unfortunately, there are some peculiar cases that could not be solved in the described manner. These cases require an embedding the additional hooks into RTI and lead to partial loss of benefits considered above.

The problem of the legacy project maintenance can also be considered as an integration problem, but it deserves more detailed consideration. HLA defines a set of common service devoted to a wide range of simulation tasks. This service set is redundant and inconvenient to be used with the usual DRE simulations whereas the absence of functionalities that could be useful in this particular task.

Despite all the reasoning related to common integration problems remains relevant, the legacy project maintenance problem could be solved with use of another principle. There could be some STAND subsystems requiring renewal, besides its runtime. Thereby if there exists any programming language which is acceptable to DRE simulation, there is a sense in constructing the HLA-compliant binding for this language and develop an additional translator for old projects.

In summary, the replacement of the current STAND runtime with the concept designed reduces to the following problem set:

1. Replacement of RTI architecture with more complex and productive one;

2. Development of the acceptable interface wrappers for other subsystems;

3. Injecting of some additional functionality and required low-level services.

## VI. Conclusion

Using of the HLA distributed simulation standard for building DRE simulation systems gives a certain benefits to the developers, namely, automatic ability to execute any HLA-compliant models and to participate in distributed simulation. Building of the DRE simulation runtime raises a number of development problems. The specific of DRE simulation imposes some additional requirement to the runtime, and specifications of the HLA standard do not satisfy the appropriate product. There are two possible solutions: addition of QoS policies to existing CERTI implementation and using of the HLA standard over the DDS standard. The considered solutions were analyzed and a second one was chosen.

Due to existence of some more or less suitable turnkey RTI, the upcoming development promises to be easier than the development from scratch. It reduces to refinement of the RTI architecture, injecting the RTI with some new functionality and developing of interface wrappers.

## References

[1] R.L. Smeliansky, A.G. Bakhmurov, and V.A. Kostenko, "DYANA - an environment for simulation and analysis of distributed multiprocessor computer systems," Moscow State University, Computational Math. and Cybern. Dept., 1999.

[2] V.V. Balashov et al., "A hardware-in-the-loop simulation environment for real-time systems development and architecture evaluation," in International Conference on Dependability of Computer Systems, 2008, pp. 80-86.

[3] R.G. Sargent, "Requirements of a modeling paradigm," in Winter Simulation Conference, Arlington, USA, 1992, pp. 780- 782.

[4] E.H. Page, "Simulation modeling methodology: principles and etiology of decision support," Department of Computer Science, Virginia Tech, Blacksburg, USA, Ph.D. Dissertation 1994.

[5] R.E. Nance, A history of discrete event simulation programming languages. Blacksburg, USA, 1993.

[6] E. Noulard, J.Y. Rousselot, and P. Siron, "Spring Simulation Interoperability Workshop," in CERTI, an open source RTI, why and how, San Diego, USA, 2009.

[7] Object Management Group; Object Interface Systems, Inc; Real-Time Innovations, Inc; THALES, Data Distribution Service for Real-time Systems, version 1.2., 2007.

[8] M. Adelantado, P. Siron, and Chaudron J.B., "Towards an HLA run-time infrastructure with hard real-time capabilities," in International Simulation Multi-Conference, Ottava, Canada, 2010.

[9] Real-Time Innovations, Inc. (RTI), "OMG Data-Distribution Service (DDS): architectural overview," 2004.

[10] Richard D. Fujimoto, Parallel and distributed simulation systems, 2000.

[11] IEEE Std 1516.1-2000, "IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA) - federate Interface specification," 2001.

[12] L. Bononi, M. Bracuto, D'Angelo G., and Donatiello L., "A new adaptive middleware for parallel and distributed Simulation of dynamically interacting systems," in Distributed Simulation and Real-Time Applications, 2004, pp. 178 - 187.

[13] I. Birrer, B. Carnicero-Dominguez, M. Egli, B. Carnicero-Dominguez, and A. Pasetti, "EODiSP – an open and distributed simulation platform," in International Workshop on Simulation for European Space Programmes, Noordwijk, the Netherlands, 2006.

[14] P. Bieber, D. Raujol, and P. Siron, "Security architecture for federated cooperative information systems," in Annual Computer Security Applications Conference, New Orleans, USA, 2000.

[15] M. Adelantado, J.L. Bussenot, J.Y. Rousselot, P. Siron, and Betoule M., "HP-CERTI: towards a high performance, high availability open source RTI for composable simulations," in Fall simulation interoperability workshop, Orlando, USA, 2004.

[16] B. d'Ausbourg, P. Siron, and E. Noulard, "Running real time distributed simulations under Linux and CERTI," in European Simulation Interoperability Workshop, Edimburgh, Scotland, 2008.

[17] L. Malinga and WH. Le Roux, "HLA RTI performance evaluation," in European Simulation Interoperability Workshop, Istanbul, Turkey, 2009, pp. 1-6.